Mike Young
Brett Ratner
Josh Saunders

S.P.I.T bike
(self powering interactive technology bike)

Description of the project:

Our smart bike is meant to bring technology to a common bicycle to make it a more functional means of transportation. The primary feature of our bike is its turn signals. At the push of a button, LEDs light up on the back of the bike, drawing an arrow pointing in the direction the user chooses. Lights on the handlebars also notify the user and anyone in front of the bike of the direction. After the bike turns in that direction, the lights will automatically turn off when the bike continues straight. The Arduino computer that controls these functions is powered by a rechargeable Lithium Ion Battery mounted on a PowerBoost 500 Shield. A pedal power generator allows us to harness power from the back wheel as it spins in order to recharge the battery.

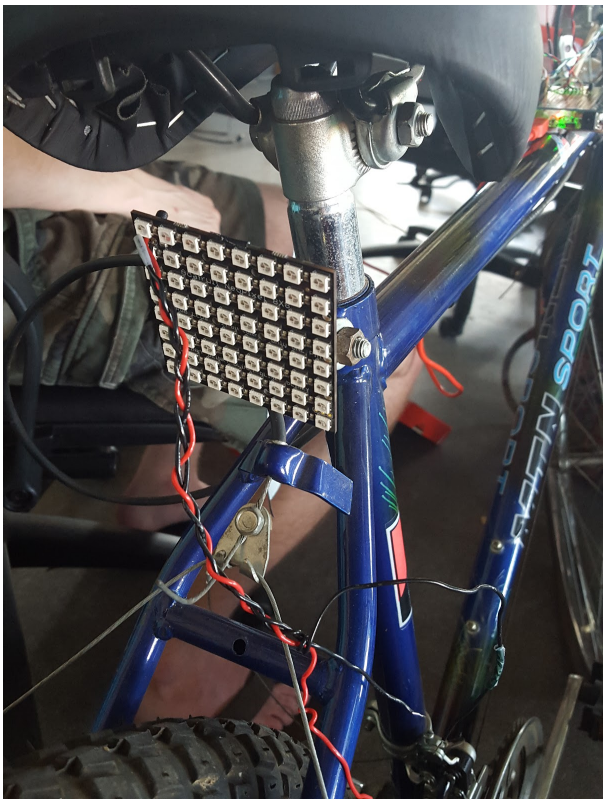
Research

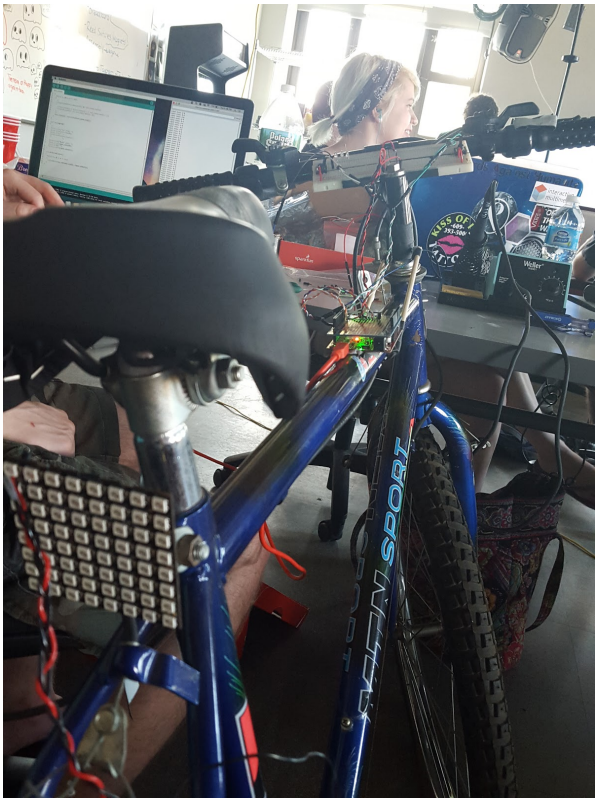
http://www.instructables.com/id/turn-signal-biking-jacket/ This is a similar project that incorporates wearable technology. While this may have its advantages, we preferred the idea of attaching all of the physical computing to the bicycle. Now you don't need to wear the same outfit every night you want to bike. Ours also incorporates charging the battery through riding.

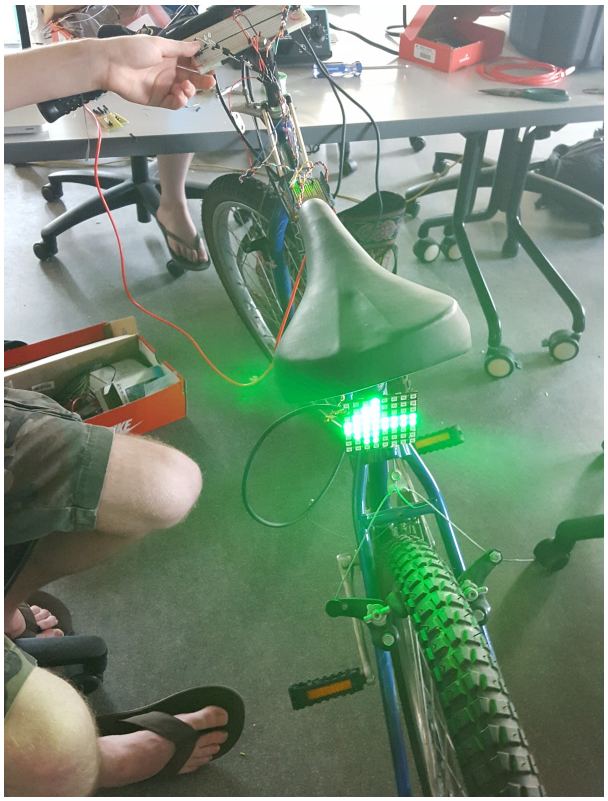
http://organictransit.com This project uses a simialr system for a self powered bike that goes above and beyond with many features that normally bikes dont have.

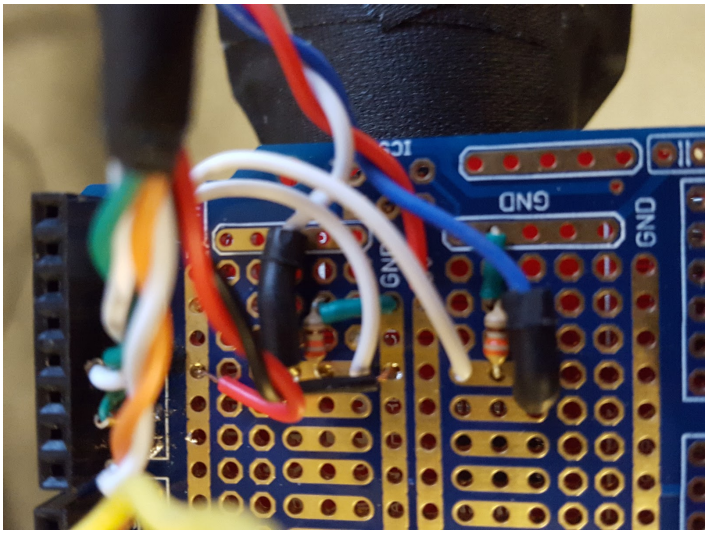
Documentation/Images:

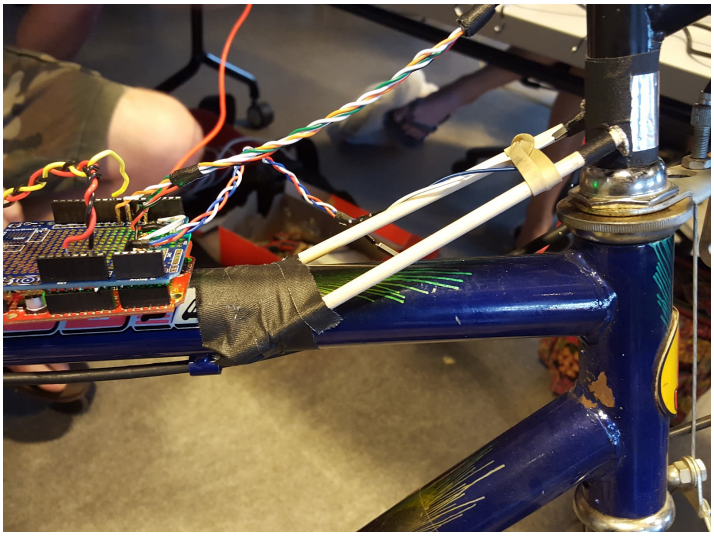
[Click here for video of smart bike in action]

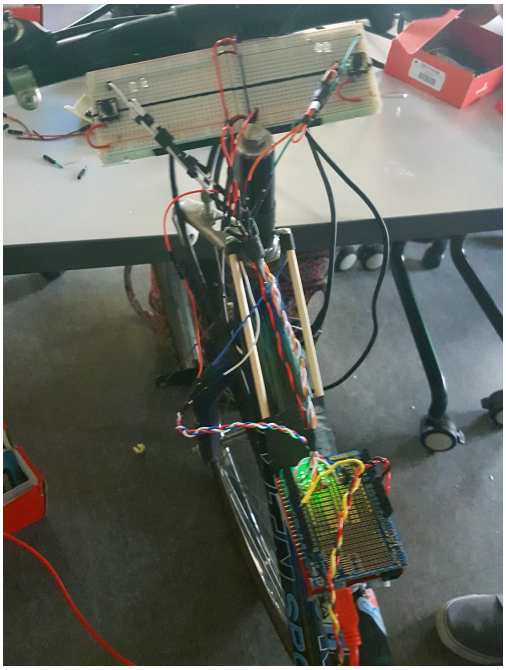




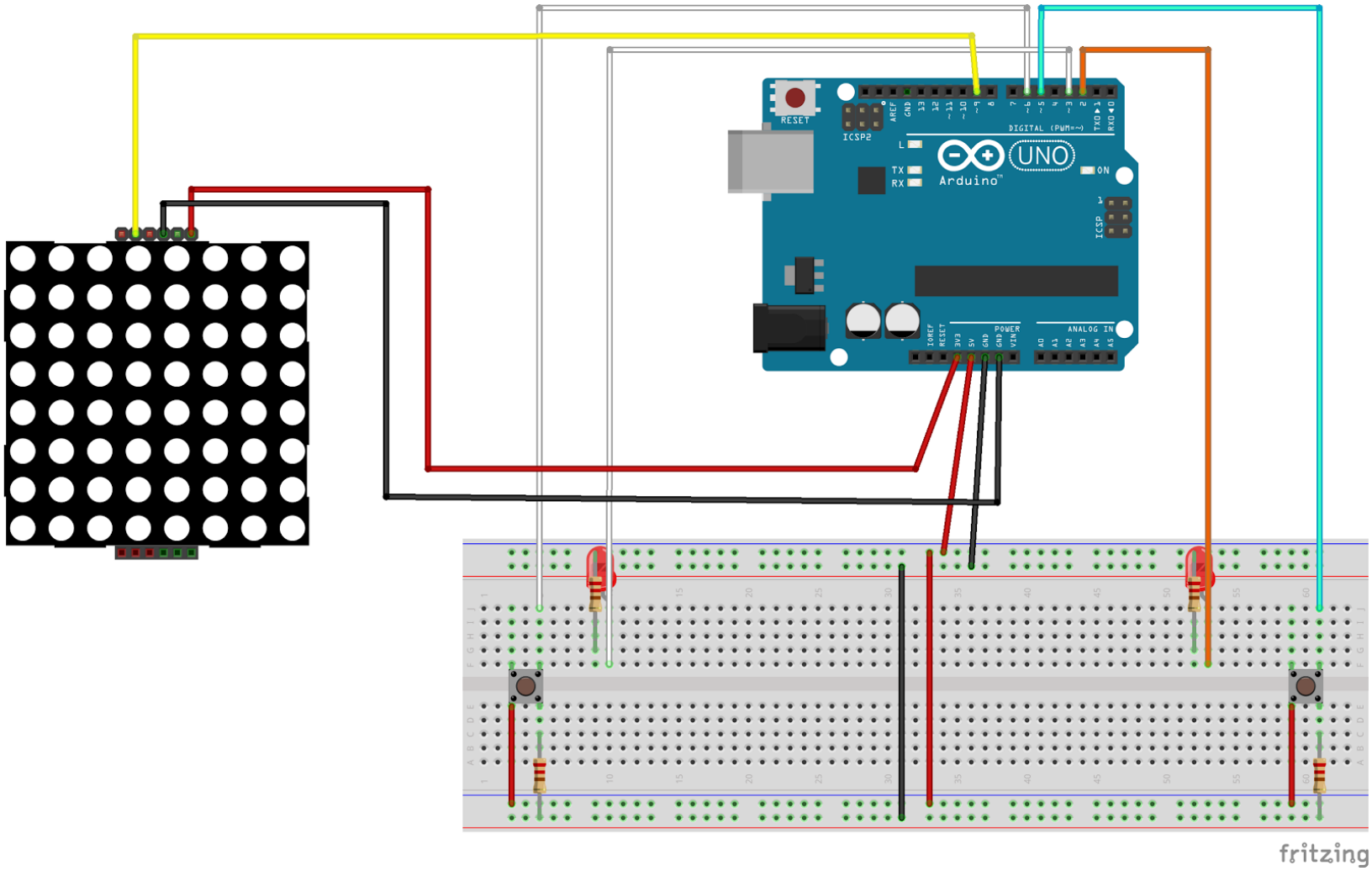



GND
GND





Fritzing Diagram s:



Source code:

```
// neopixel
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>
#define PIN            9
#define NUMPIXELS      64
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
int led_matrixRight[] = {0,0,0,0,1,0,0,0,
                         0,0,0,0,1,1,0,0,
                         0,0,0,0,1,1,1,0,
                         1,1,1,1,1,1,1,1,
                         1,1,1,1,1,1,1,1,
                         0,0,0,0,1,1,1,0,
                         0,0,0,0,1,1,0,0,
                         0,0,0,0,1,0,0,0};

int led_matrixLeft[] = {0,0,0,1,0,0,0,0,
                        0,0,1,1,0,0,0,0,
                        0,1,1,1,0,0,0,0,
                        1,1,1,1,1,1,1,1,
                        1,1,1,1,1,1,1,1,
                        0,1,1,1,0,0,0,0,
                        0,0,1,1,0,0,0,0,
                        0,0,0,1,0,0,0,0};

// pins
const int turn_sense[] = {2, 3}, blinker_btn[] = {5, 6}, blinker_led[] = {7, 8};

// states
int button_state[] = {0, 0}, turn_state[] = {0, 0}, use_delay = 0;
int state;
```

```
void setup()
{
  #if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
  #endif

  Serial.begin(9600);

  Serial.print("initializing neopixels...");
  pixels.begin();
  Serial.println("OK");

  Serial.print("initializing pins...");
  for(int x = 0; x <= 1; x++)
  {
    pinMode(turn_sense[x], INPUT);
    pinMode(blinker_btn[x], INPUT);
    pinMode(blinker_led[x], OUTPUT);
  }
  Serial.println("OK");
}

void loop()
{
  run_turns();
  run_blinker();
  logging();
  if(use_delay){delay(500); use_delay = 0;}
}

void logging()
```

```
{
  /*
  Serial.print("btn state: ");
  Serial.print(button_state[0]);
  Serial.print(" ");
  Serial.println(button_state[1]);
  */

  Serial.print("turn state: ");
  Serial.print(turn_state[0]);
  Serial.print(" ");
  Serial.println(turn_state[1]);
}

void read_blinkers()
{
  int blinker_btn_state[2];
  for(int x = 0; x <= 1; x++){blinker_btn_state[x] = digitalRead(blinker_btn[x]);}

  if(blinker_btn_state[0] && blinker_btn_state[1]){button_state[0] = 0; button_state[1] = 0; use_delay = 1;}
  else if(blinker_btn_state[0] || blinker_btn_state[1])
  {
    for(int x = 0; x <= 1; x++)
    {
      if(blinker_btn_state[x] && button_state[x] != 1){button_state[x] = 1; use_delay = 1;}
      else if(blinker_btn_state[x] && button_state[x] == 1){button_state[x] = 0; use_delay = 1;}
      else{button_state[x] = 0;}
    }
  }
}

void read_turns()
```

```
{
  for(int x = 0; x <= 1; x++)
  {
    if(button_state[x] && turn_state[x] && !digitalRead(turn_sense[x]))
    {
      button_state[0] = 0;
      button_state[1] = 0;
    }
    else
    {
      turn_state[x] = digitalRead(turn_sense[x]);
    }
  }
}

void run_blinker()
{
  read_blinkers();

  if(button_state[0]){state = 2;}
  else if(button_state[1]){state = 1;}
  else{state = 0;}
  tail_light(0); // reset
  tail_light(state);
}

void run_turns()
{
  read_turns();
}

void tail_light(int state)
```

```
{
  for(int i=0;i<NUMPIXELS;i++)
  {
    switch(state)
    {
      case 1:
        if(led_matrixRight[i] == 1){pixels.setPixelColor(i, pixels.Color(0,100,0));}
        break;
      case 2:
        if(led_matrixLeft[i] == 1){pixels.setPixelColor(i, pixels.Color(0,100,0));}
        break;
      default:
        pixels.setPixelColor(i, pixels.Color(0,0,0));
    }

    pixels.show();
  }

  if(state == 1 || state == 2)
  {
    if(state == 1){digitalWrite(blinker_led[0], HIGH); digitalWrite(blinker_led[1], LOW);}
    else{digitalWrite(blinker_led[0], LOW); digitalWrite(blinker_led[1], HIGH);}
    delay(500);
  }
  else
  {
    digitalWrite(blinker_led[0], LOW);
    digitalWrite(blinker_led[1], LOW);
  }
}
```